PATENT ABSTRACTS OF JAPAN

(11)Publication number : 08-278918
(43)Date of publication of application : 22.10.1996

(51)Int.Cl. G06F 12/04
G06F 9/46

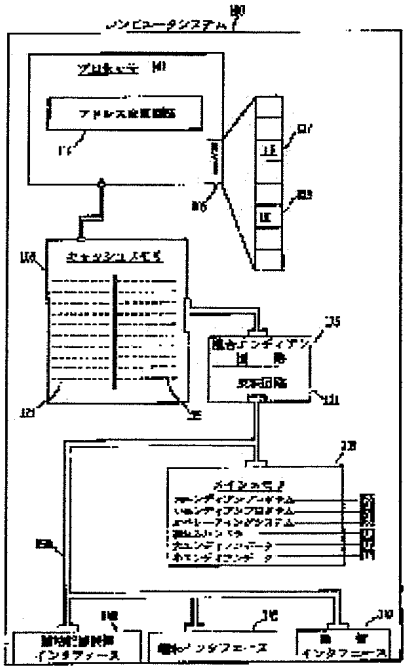
(21)Application number : 08-026648
(22)Date of filing : 14.02.1996

(71)Applicant : INTERNATL BUSINESS MACH CORP <IBM>
(72)Inventor : CARNEVALE MICHAEL J
HOPKINS MARTIN EDWARD
LOEN LARRY WAYNE
SILHA EDWARD JOHN
ANDREW HENRY WOTTRENG

(30)Priority
Priority number : 95 394072 Priority date : 24.02.1995 Priority country : US

(54) SYSTEM AND METHOD FOR EXECUTING ENDIAN TASK

(57)Abstract:
PROBLEM TO BE SOLVED: To provide a hybrid Endian computer system supporting a task having a different information form by one computer system.
SOLUTION: The computer system 100 is provided with a hybrid Endian circuit 175 to revise its Endian mode dynamically and to enhance the function of a conventional 2 Endian computer system. The hybrid Endian computer system 100 revises the Endian mode for each task as required. The hybrid Endian circuit 175 formats data automatically in a form expected by a task in execution independently of whether or not the task expects data in a large Endian format or a small Endian format. The hybrid Endian circuit 175 formats data so that a large Endian instruction or a small Endian instruction is executed by one and same computer system.



(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号

特開平8－278918

(43)公開日　平成８年(1996)10月22日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ	12/04	５１０		Ｇ０６Ｆ	12/04	５１０Ｇ
	9/46	３４０			9/46	３４０Ｚ

審査請求　未請求　請求項の数20　ＯＬ　（全 21 頁）

(21)出願番号　特願平8－26648

(22)出願日　平成８年(1996)２月14日

(31)優先権主張番号　３９４０７２

(32)優先日　1995年２月24日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　390009531
インターナショナル・ビジネス・マシーンズ・コーポレイション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＳＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国10504、ニューヨーク州　アーモンク　（番地なし）

(72)発明者　マイケル・ジョゼフ・カーネベイル
アメリカ合衆国ミネソタ州、ロチェスタ、18　１／２ストリート　エヌ・ダブリュ　2110

(74)代理人　弁理士　合田　潔　（外２名）

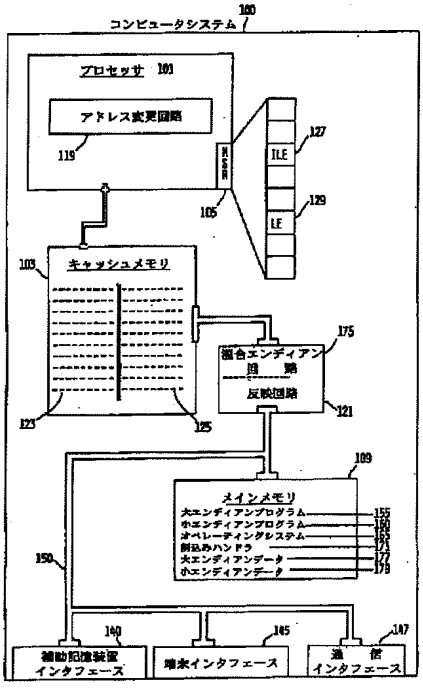
最終頁に続く

(54)【発明の名称】　エンディアンタスクを実行するシステム及び方法

(57)【要約】

【課題】　１つのコンピュータシステムで異なる情報形式を有するタスクをサポートする、混合エンディアンコンピュータシステムの提供。

【解決手段】従来の２エンディアンコンピュータシステムの機能を高め、前記コンピュータシステムにそのエンディアンモードを動的に変更することを可能にする混合エンディアン回路を備えるようにする。混合エンディアンコンピュータシステムは、必要なら、タスク毎にエンディアンモードを変更できる。混合エンディアン回路は、タスクが大エンディアンフォーマット又は小エンディアンフォーマットのデータを期待するかどうかに関係なく、実行中のタスクが期待する形式で自動的にデータをフォーマット化する。混合エンディアン回路はまた大エンディアン命令又は小エンディアン命令が同じコンピュータシステムで実行できるようにそれらをフォーマット化する。



1

【特許請求の範囲】

【請求項１】プロセッサと、

前記プロセッサで、タスク毎に、直接に、大エンディアンタスクを実行する大エンディアンプログラム及び小エンディアンタスクを実行する小エンディアンプログラムを含むメモリとを備えるコンピュータシステム。

【請求項２】前記メモリは、前記大エンディアンタスクによりアクセスされる大エンディアンデータ及び前記小エンディアンタスクによりアクセスされる小エンディアンデータを更に含む、請求項１に記載のコンピュータシステム。

【請求項３】前記小エンディアンデータは前記大エンディアンタスクによりアクセスされ、そして前記大エンディアンデータは前記小エンディアンタスクによりアクセスされる、請求項２に記載のコンピュータシステム。

【請求項４】前記メモリは前記小エンディアンデータ及び前記大エンディアンデータを記憶装置から受取る、請求項３に記載のコンピュータシステム。

【請求項５】前記メモリは前記小エンディアンデータ及び前記大エンディアンデータを通信ネットワークから受取る、請求項３に記載のコンピュータシステム。

【請求項６】前記メモリは前記コンピュータシステムにバス接続されている別のコンピュータシステムから前記小エンディアンデータ及び前記大エンディアンデータを受取る、請求項３に記載のコンピュータシステム。

【請求項７】前記大エンディアンプログラムは大エンディアンフォーマットの命令を含み、そして前記小エンディアンプログラムは小エンディアンフォーマットの命令を含む、請求項１に記載のコンピュータシステム。

【請求項８】少なくとも１つの２エンディアンプロセッサを有する複数のプロセッサと、

大エンディアンタスク及び小エンディアンタスクがどちらも前記少なくとも１つの２エンディアンプロセッサで、タスク毎に、直接に、実行されるとき、前記大エンディアンタスクを前記プロセッサで実行する大エンディアンプログラム及び前記小エンディアンタスクを前記プロセッサで実行する小エンディアンプログラムを含むメモリとを備え、前記メモリは、前記少なくとも１つの２エンディアンプロセッサの代わりのエンディアンモードでデータを取出す間に、前記大エンディアンタスク及び前記小エンディアンタスクが前記少なくとも１つの２エンディアンで行われるデータ及びアドレス調整を受けないように、一貫性のあるメモリイメージを提供する多重プロセッサコンピュータシステム。

【請求項９】前記データは、前記大エンディアンタスクでアクセスされる大エンディアンデータ及び前記小エンディアンタスクでアクセスされる小エンディアンデータを含む、請求項８に記載のコンピュータシステム。

【請求項１０】前記データは、前記小エンディアンタスクでアクセスされる大エンディアンデータ及び前記大エンディアンタスクでアクセスされる小エンディアンデータを含む、請求項８に記載のコンピュータシステム。

2

【請求項１１】従来の２エンディアンプロセッサで行われるデータ及びアドレス調整を追跡し且つ制御する混合エンディアン機構を備えるコンピュータシステム。

【請求項１２】前記従来の２エンディアンプロセッサは大エンディアンバイアスされ、そして前記従来の２エンディアンプロセッサが小エンディアンデータを処理できるように前記データ及びアドレス調整が行われる、請求項１１に記載のコンピュータシステム。

【請求項１３】前記従来の２エンディアンプロセッサは小エンディアンバイアスされ、そして前記従来の２エンディアンプロセッサが大エンディアンデータを処理できるように前記データ及びアドレス調整が行われる、請求項１１に記載のコンピュータシステム。

【請求項１４】１つのプロセッサで直接に実行するために大エンディアンタスクを開始するステップと、

前記プロセッサで直に実行するために前記大エンディアンタスクに割込み、そして小エンディアンタスクを直ちに開始するステップとを含むコンピュータでエンディアンタスクを実行する方法。

【請求項１５】前記大エンディアンタスクは大エンディアンデータをアクセスし、そして前記小エンディアンタスクは前記小エンディアンデータをアクセスする、請求項１４に記載の方法。

【請求項１６】前記大エンディアンタスクは小エンディアンデータをアクセスし、そして前記小エンディアンタスクは大エンディアンデータをアクセスする、請求項１５に記載の方法。

【請求項１７】補助記憶装置からメモリに前記小エンディアンデータ及び前記大エンディアンデータを受取る、請求項１６に記載の方法。

【請求項１８】外部通信ネットワークからメモリに前記小エンディアンデータ及び前記大エンディアンデータを受取る、請求項１６に記載の方法。

【請求項１９】前記コンピュータシステムにバス接続されている別のコンピュータシステムからメモリに前記小エンディアンデータ及び前記大エンディアンデータを受取る、請求項１６に記載の方法。

【請求項２０】前記大エンディアンタスクは大エンディアンフォーマットの命令を含む大エンディアンプログラムの複数のバージョンを実行し、そして前記小エンディアンタスクは小エンディアンフォーマットの命令を含む小エンディアンプログラムの複数のバージョンを実行する、請求項１４に記載の方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は一般にデータ処理システムに、より詳しくは、エンディアン指向コンピュータシステムに関する。

【０００２】

【従来の技術】1940年代後期に製造された最も初期のディジタルコンピュータは簡単な設計及び構成素子を有した。長年にわたる技術の多くの進歩にもかかわらず、大抵の今日のコンピュータは情報の記憶及び処理について依然として同じ基本的な素子を用いて同じ基本的なタスクを実行する。

【０００３】これらの基本的な素子のうちの２つはコンピュータメモリ及びプロセッサである。コンピュータメモリはそのコンピュータで用いられる情報を記憶し、そして多くの点で人間の記憶と同じように働く。例えば、人々が異なる話題及び事象に関する着想を記憶できるのと全く同じように、コンピュータシステムのメモリは文字、数字、絵及びその他の形式の情報を記憶するのに使用できる。

【０００４】コンピュータプロセッサはコンピュータシステムの能動素子である。プロセッサはコンピュータシステムのメモリに記憶された情報を操作し、コンピュータシステムに割当てられたタスクを実行する。コンピュータシステムにより処理されるタスクは時にはジョブ、プログラム又はプロセスとも呼ばれる。

【０００５】ジョブを実行するコンピュータプロセッサは、コンピュータメモリに記憶された情報を、多くの点で、人間が本のページに印刷された文字を読んで処理するのと同じように、読取って処理する。それゆえ、ページ上の文字の配列がそれを読む人にとって重要であるのと全く同じように、コンピュータシステムのメモリでの情報の配列はコンピュータシステムにとって重要である。例えば、英語の文字は左から右に書かれ、そしてヘブライ語の文字は右から左に書かれる。英語のみを読む人々は左から右に書かれる英語の文字を理解でき、そしてヘブライ語のみを読む人々は右から左に書かれるヘブライ語の文字を理解できる。英語の文字がページ上で左から右に書かれることは、文字を、右から左に配列する代わりに、このように配列する方がよいことを意味しない。英語の文字が左から右に書かれるのは、単に、文字が左から右に書かれることを英語の読者が期待するからである。この意味で、コンピュータシステムのメモリ内の情報の配列はページ上の文字の配列と異なるものではない。情報を配列する１つの方法は情報を配列する他の任意の方法よりもすぐれてはいない。しかしながら、もし情報がコンピュータシステムが期待する方法で配列されなければ、コンピュータシステムは情報を理解できないという意味では、コンピュータシステムも人々と全く同様である。

【０００６】もちろん、１つのコンピュータシステム情報配列方法が他の任意の情報配列方法よりもすぐれてはいないことは、コンピュータシステムメモリ内での情報の構成に対する異なるアプローチに〝門戸を開放〟している。そして、コンピュータシステム設計者が少なくとも幾つかの形式のコンピュータシステム情報を構成する異なる方式をいつか開発するであろうことは驚くにあたらない。少し以前に１つの前記変化がコンピュータシステム情報の２つの特定の形式（浮動小数点情報及び２進整数情報と呼ばれる）について起きた。現在、これらのタイプのコンピュータシステム情報を配列する２つの共通の方式がある。この２つの方式はJonathan Swiftの有名な作品〝ガリヴァー旅行記〟の相容れない部族の名をとって〝小エンディアン〟及び〝大エンディアン〟と呼ばれた。用語は派手であるが、それらはコンピュータシステムにとっては、これらのタイプの情報が特定のコンピュータシステムに配列される方法を示すこと以外は重要ではない。

【０００７】1970年代の後期及び1980年代の初期に、インテル社が導入したプロセッサは、IBM PC及び互換性パーソナルコンピュータの標準になった。これらのコンピュータシステムはいわゆる小エンディアン配列を用いた。これと同じ時機に、いわゆる大エンディアン配列を用いる他のコンピュータシステムが設計された。これらの後のコンピュータシステムはモトローラ社製のプロセッサを備え、そしてアップル社製のコンピュータシステムで用いられた。

【０００８】以前は、１つのコンピュータシステムに２つ以上の方法で情報を配列することは利点とは考えられなかったので、コンピュータシステムがそれらのメモリ内に（大エンディアン又は小エンディアン形式で）情報を構築した方法は重要な問題ではなかった。

【０００９】しかしながら、今日、業務及び家庭で用いられるコンピュータの巨大な成長は異なるタイプのコンピュータシステムの間の互換性の深刻な必要性を生じている。例えば、一般にIBM PC又は互換性のコンピュータを用いる人はアップル社のマッキントッシュコンピュータを用いる人とプログラム及び情報を共有できない。その逆の場合もある。両タイプのコンピュータを用いる大会社は従業員間の情報の分配に困難を生じている。小会社は同じタイプのコンピュータを保有しない供給者又は購買者と容易に情報を共有できないことがよくある。その結果、コンピュータソフトウェア開発者は、異なるタイプのコンピュータシステムをサポートするために同じソフトウェアの複数のバージョンを開発する余分な時間及び資源の投入をしばしば強制される。要するに、１つのコンピュータシステムにおいて一定のタイプの情報を２つ以上の方法で配列する能力の欠如は大体において生産の遅延、生産性の下落、効率の低下及び資本経費の増大をもたらす。

【００１０】既存のコンピュータシステムのなかには、限定的に過ぎないが、エンディアン問題の処理を試みるものがある。これらのコンピュータシステムは２エンディアン機能と呼びうるものを備える。本質的には、２エンディアン機能は、同じコンピュータシステムが大エン

ディアンタスク又は小エンディアンタスクのどちらかを実行できるが、両者を同時には実行できないことを意味する。すなわち、コンピュータシステムは、先ず始動されたとき、大エンディアンタスクを実行するか又は小エンディアンタスクを実行するかを問われる（即ち、コンピュータシステムは大エンディアン又は小エンディアンのどちらのモードでも実行できるようにしうる）。既存の２エンディアンコンピュータシステムのエンディアンモードの切換え（即ち、大エンディアンモードから小エンディアンモードに、又はその逆の切換え）は、コンピュータの初期化サイクルで極めて早期に（即ち、始動に極めて接近して）実行する特別な目的のソフトウェアを必要とする。その後、全てのタスクは指定されたエンディアンで実行する。そして、実際には、２エンディアンコンピュータシステムは、それらがひとたびそれらのブートプロセスを終了したならば、通常の〝単エンディアン〟コンピュータシステムに全く等しい。

【００１１】これらの２エンディアンコンピュータシステムは、大エンディアン又は小エンディアンのタスクのどちらも実行する能力を提供することにより、更に柔軟性を与えるが、それらは、１つのコンピュータシステムで、異なる情報形式期待（即ち、大エンディアン対小エンディアン）を有するタスクの共存及びタスク毎に可能な実行の要求に関連した問題を解決するものではない。

【００１２】小エンディアン環境で生成されたプログラム、及び大エンディアン環境で生成された他のプログラムを同時に実行できるコンピュータシステムなしには、今日の非互換性問題はコンピュータ産業を苦しめ続けるであろう。

【００１３】

【発明が解決しようとする課題】本発明の第１の目的は、再初期化を必要とせずに異なるエンディアンのタスクをサポートする高機能コンピュータシステムを提供することにある。

【００１４】本発明の第２の目的は、１つのコンピュータシステムで、異なる（即ち、大エンディアン対小エンディアン）情報形式期待を有するタスクが共存し且つタスク毎に実行できる高機能コンピュータシステムを提供することにある。

【００１５】本発明の第３の目的は、大エンディアンデータ又は小エンディアンデータを１つのコンピュータシステムに記憶し、それを大エンディアンデータ又は小エンディアンデータを期待するタスクがそれぞれアクセスできる、高機能コンピュータシステムを提供することにある。

【００１６】本発明の第４の目的は、１つのコンピュータシステムで、異なる（即ち、大エンディアン対小エンディアン）情報形式期待を有するタスクが共存し且つタスク毎に実行すると同時に、代わりのエンディアン形式でフォーマット化されたデータをそれぞれアクセスできる、高機能コンピュータシステムを提供することにある。

【００１７】本発明の第５の目的は、１つのコンピュータシステムで、異なる（即ち、大エンディアン対小エンディアン）情報形式期待を有するタスクが共存し且つタスク毎に実行すると同時に、しかも多重プロセッサ環境で一貫性のあるメインメモリイメージを供給できる、高機能コンピュータシステムを提供することにある。

【００１８】

【課題を解決するための手段】１つのコンピュータシステムで、異なる（即ち、大エンディアン対小エンディアン）情報形式期待を有するタスクが共存し且つタスク毎に実行できる新規の装置及び方法が本明細書で開示される。以降、このコンピュータシステムは混合エンディアンコンピュータシステムと呼ばれる。

【００１９】本発明の混合エンディアンコンピュータシステムは既存の２エンディアンコンピュータシステムの拡張である。コンピュータシステムがそのエンディアンモードを動的に変更できる混合エンディアン回路が付加される。混合エンディアンコンピュータシステムは必要ならタスク毎にエンディアンモードを変更できる。混合エンディアン回路は、タスクが大エンディアンフォーマットのデータを期待するか又は小エンディアンフォーマットのデータを期待するかに関係なく、自動的に、実行中のタスクにより期待された形式にデータをフォーマット化する。また、混合エンディアン回路は、大エンディアン命令及び小エンディアン命令を、同じコンピュータシステムで実行できるようにフォーマット化する。

【００２０】

【発明の実施の形態】

［概説］前述のように、本発明は１つのコンピュータシステムでの大（ビッグ）エンディアン情報及び小（リトル）エンディアン情報の記憶及び使用に関する。より詳しくは、本発明は、異なるエンディアン（即ち、大エンディアン又は小エンディアン）期待を有するタスクが１つのコンピュータシステムに共存するのみならず、１つのコンピュータシステムでタスク毎に実行することもできる、高機能コンピュータシステムである。

【００２１】本発明の利点及び長所を読者が完全に理解できるように問題及び解法の基本的な概要をここに示す。本発明の内部の働きの詳細な記述は本明細書の［詳細な説明］の項に記載される。

【００２２】図３に示すように、大エンディアンデータフォーマットは、最下位メモリアドレスに、ワードの最上位バイトを、最下位プラス１のメモリアドレスに、ワードの次の最上位バイトを記憶する必要がある。以下同様である。例えば、50,000の10進値を表わす32ビット整数は16進数の0000C350として昇順バイトで記憶される。ここで、アドレスオフセット0は00であり、そしてオフセット3は50である。対照的に、小エンディアンフォー

マットは、最下位バイトの記憶が最下位メモリアドレスに、次の最下位バイトは最下位プラス１のメモリアドレスに記憶されることを必要とする。以下同様である。例えば、50,000の10進値を表わす32ビット整数は16進数の50C30000として昇順バイトで記憶される。ここで、アドレスオフセット0は50であり、そしてオフセット3は00である。

【００２３】エンディアン問題のために、小エンディアンマシン上で走行するように記述されたコンピュータプログラムは一般に大エンディアンマシン上では走行しない。その逆も同じである。例えば、最初に１つのエンディアンのためにコンパイルされ、そして他のエンディアンのためにコンパイルし直されたときは、たとえ完全に標準化された言語が用いられるときでも、無変更で動作するプログラムの設計は現在は困難である。これは、言語標準がデータに関してエンディアンを指定せず、そしてプログラマはコンパイルするコンピュータシステムの基礎をなすエンディアンを常に用いるためである。プログラマはしばしば所与の記憶片を代わりの定義で重ね書きするから、下記の表（以降、表中に現れる英文字の列はコマンドなどを表す記号であり、翻訳できない）のＣコードのセグメントが示すように、基礎をなすエンディアンが異なるマシンのために変化するとき、記憶を再定義するソースコードは同じ結果を生ずるように変更せねばならないことがある：

【表１】

```
typedef struct {
        int a;
        short int b;
        char c[5];
} example;

example x:
x.a = 0x01020304;
x.b = 0x1112;
strcpy(x.c,"\x21\x22\x23\x24");
```


【００２４】小エンディアンコンピュータでは、情報はメモリに16進数として下記のように記憶される：

【表２】

Offset	0	1	2	3	4	5	6	7	8	9	A
	04	03	02	01	12	11	21	22	23	24	00

【００２５】大エンディアンコンピュータでは、情報はメモリに16進数として下記のように記憶される：

【表３】

Offset	0	1	2	3	4	5	6	7	8	9	A
	01	02	03	04	11	12	21	22	23	24	00

【００２６】ここで下記について考慮する：

【表４】

```
INT64 timestamp_whole;
typedef struct {
        INT32 upper,    /* 秒単位の時間 */
        INT32 lower,    /* 下位32ビットの時間(秒の端数) */
} timestamp_split;
timestamp_split X = *(&(timestamp_split) timestamp_whole);
timestamp New;
New = X;
```

【００２７】データ構造 "timestamp_split"は大エンディアンである。なぜなら、それは２つの32ビット整数により64ビット整数に重ね書きするからである。変数Ｘ及びNew の設定のみが大エンディアンの規則の下に正しく働く。上位の32ビットが秒単位の時間である場合、多くの時間のスタンプフォーマットで、その秒をアクセスする必要性は便宜的に前記構造を必要とする。しかしながら、プログラムが小エンディアンのタイムスタンプとともに小エンディアン環境に移る場合、このデータ構造の参照はどれもアプリケーションを通じて変更する必要がある。これは、上記のプログラムが大エンディアンの順序付けを想定してダブルワード内のワードを絶対的に順序付けるからである。そしてこの順序付けは、たとえ小エンディアンの目的マシンのために再コンパイルされた場合でも、同じ順序のままである。なぜなら、上記の表４で "timestamp_split" は形式的に２つの隣接する４バイト整数としての順序付けを記述するからである。

【００２８】それはこの種の順序付け関係の符号化であり、多くの外観を有するので、プログラムはそのプログラムの最初の目標にされるエンディアンに広く分散した従属状態を有する。予め注意深く計画されない限り、１つのエンディアン環境を有するシステムで開発されたプログラムは他のエンディアン環境を有するシステムに移植するすることは困難であり且つ費用がかかる。よって、１つのエンディアン環境を用いるシステムで実行するために設計されたプログラムは他のエンディアン環境のコンピュータシステムで実行するために変換されることは稀である。

【００２９】説明のために、この概要を通じて32ビットのワードサイズが用いられ、そして対応するワードの部分が図３に示される。しかしながら、他のワードサイズ、例えば16ビット及び64ビットを用いるコンピュータ

システムは、本発明により一般性を失わずに使用できることが分かる。

【００３０】前述のように、あるコンピュータシステムは、２エンディアンコンピュータシステムと呼ばれ、それらの外部のエンディアンモードを変更できる。しかしながら、内部的には、２エンディアンコンピュータシステムは１つの特定のエンディアンの方にバイアスされるコンピュータシステムとみなしうるが、同時に代わりのエンディアンモードを提供する回路を有する。この回路は、そのコンピュータシステムのエンディアンモードを制御するために特別な目的のソフトウェアにより初期化される。代わりのエンディアンモードで動作するようにこの回路が初期化されると、データの反映が起きる。反映の必要性は単に、２エンディアンコンピュータシステムが大エンディアン及び小エンディアンのデータフォーマットの間の相違をいかに処理するかの結果である。それゆえ、ひとたび２エンディアンコンピュータシステムが始動され走行すれば、もちろん、そのコンピュータシステムがその代わりのエンディアンモードで走行するように告げられているかどうかにより、反映は常に起きるか又は決して起きない。

【００３１】本発明の良好なプロセッサは高機能IBM パワーPCプロセッサである。パワーPCアーキテクチャは PowerPC Architecture Manual, May, 1993, IBM Corporation, Pub. No. SR28-5124-00 に記述されている。パワーPCの２エンディアンプロセッサ（例えば、モデル620）ファミリは大エンディアンの方にバイアスされる。パワーPC２エンディアンプロセッサは良好な実施例として選択されているが、本発明はどの特定の２エンディアンプロセッサにも且ついかなる特定のエンディアンバイアスにも限定されないことを当業者は理解するであろう。

【００３２】IBM ２エンディアンパワーPCプロセッサは大エンディアンの方にバイアスされるので、それらは２ステッププロセスを介して小エンディアンタスクを処理する。第１のステップは前述の反映である。反映ステップは本質的にはデータを再配列し、それが所与のプロセッサの大エンディアンバイアスを正しくアクセスできるようにする。第２のステップは、アドレス変更と呼ばれ、大エンディアンアドレスから小エンディアンアドレスにデータを参照するために用いられるアドレスを変換する。この点で、繰り返して言えば、本発明はいかなる特定のエンディアンバイアスにも限定されないことである。従って、小エンディアンバイアスのために、あるタイプの反映及びアドレス変更を通じて大エンディアンタスクを処理する２エンディアンプロセッサのどれにも、本発明は等しく適用されることが理解されるべきである。

【００３３】前述のように、混合エンディアン処理環境で走行するときに存在する２つの可能なエンディアン状況がある。第１の可能性はプロセッサの内部のエンディアンがソフトウェアタスクのエンディアンに一致する場合である。このケースでは、ソフトウェアデータの変換は不要である。なぜなら、データバイト順位及びデータアドレスオフセットはプロセッサにより直に読取りうるからである。第２の可能性はプロセッサの内部のエンディアンがソフトウェアタスクのエンディアンと一致しない場合である。この後者のケースでは、前述の２ステッププロセスはパワーPCコンピュータシステム内で用いられる。これらのステップは図４及び図５に示されたテーブルに示されている。第１のステップはデータダブルワード又はその一部分を含むバイトで実行すべき反映である（図４参照）。 第２のステップは、第１のステップで実行された反映の後のバイトの新しいロケーションを提供するための、データダブルワードを含むバイトのメモリアドレスオフセットの変更である（図５参照）。

【００３４】反映ステップは種々の場所で実行できるので、完全に機械的であり且つ取出されるデータエレメントのサイズとは無関係である。反映は代わりのエンディアンの〝真〟のエンディアンフォーマットであると想定される記憶から開始し、その後に反映される。この反映は、第２のステップのアドレス変更前の取出し／記憶経路に沿った取出し又は記憶の一部として、本来のアドレス指定されたページ、実際のアドレス指定されたページ、又は位置合わせされたダブルワードないしその明確な一部分のような関連したキャッシュ行内のその相対的なオフセットに基づく。図４に示されたテーブルで、バイト０はバイト７と交換され、バイト１はバイト６と交換され、バイト２はバイト５と交換され、そしてバイト３はバイト４と交換される。その結果は、この時点で１つのエンディアンから他のエンディアンに記憶が変更されているが、プログラマが期待したものと異なるオフセットで存在していることである。他の、64ビットと異なるワードサイズの反映は本発明により一般性を失わずに実行できる。

【００３５】図４に示され、そして本発明の機構により実行された反映は時にはダブルワード反映と呼ばれるが、ワード〝反映〟が無条件で現われるときは必ず、それは図４に示されたダブルワード反映を指すと考えるべきである。

【００３６】第２のステップはプロセッサが参照しているデータワードのサイズによるアドレス変更を実行する。図５で、良好なシステムは、８ビットバイトメモリ参照中にはXOR(排他的論理和) 7 の演算、16ビットワードメモリ参照中にはXOR 6 の演算、32ビットワードメモリ参照中にはXOR 4 の演算、そして64ビットメモリ参照中にはXOR 0 の演算を実行する。本発明の16ビット、32ビット又は64ビットワードプロセッサのどれかの良好な実施例では、アドレスオフセット変更は、アドレスオフセットの３つの最下位ビットに関する適切なXOR 演算の実行により一般化される。

【００３７】図６～図９は小エンディアンフォーマットから大エンディアンフォーマットへの変換を下記の例で示す:

【表５】

```
typedef struct {
        INT32 word;
        INT16 hword;
        BYTE byte;
        BYTE end;
} demo;

demo x;
x.word = 0x0000C350;
x.hword = 0xFIF2;
x.byte = 0xA1;
x.end = 0x00;
```

【００３８】例えば、図８で、小エンディアンフォーマットから大エンディアンフォーマットに反映された32ビットワードが示され、そしてそのロケーションが32ビットワード取出し中にオフセット０からオフセット４に変更されている。プロセッサがメモリからの取出しを終了するために、最初にソフトウェアにより提示されたときのアドレスに関してアドレス変更が実行される。この32ビットワードの取出しのケースでは、最初のアドレスは０のオフセットを有する。プロセッサは提示されたアドレスを取ってXOR 4 を実行し、対応する大エンディアンワードをオフセット４から取出す。８ビットバイトメモリ参照中のXOR 7 の演算、16ビット半ワードメモリ参照中のXOR 6 の演算、32ビットワードメモリ参照中のXOR 4 の演算、そして64ビットメモリ参照中のXOR 0 の演算を実行することにより、第１のステップの前に小エンディアンフォーマットで開始されたのち最初に上記のように反映される記憶は、最初のオフセットを最初のソフトウェア指定のオフセットから、内部的に正しいオフセットに訂正して反映を考慮することが分かる。これらのステップはパワーPCコンピュータシステムにより隠されるから、プログラマは環境が真の小エンディアン以外の何物であるかを判別できない。８ビット、16ビット及び64ビットメモリ参照に対応する、小エンディアンフォーマットから大エンディアンフォーマットへの２ステップ変換の別の例は、図６、図７及び図９に示される。

【００３９】上記のパワーPCの２ステップ変換プロセスは部分的なデータ読取りにも適応する。例えば、図１０は32ビット整数 00 00 C3 50の下位16ビット C3 50の取出しを示す。32ビット整数は小エンディアンであるので、ソフトウェアは通常は標準の小エンディアン規則を用いてオフセット０の半ワードを取出す。32ビットワードの16ビットメモリ参照は、前に図８で説明したように、最初に32ビットワードを小エンディアンフォーマットから大エンディアンフォーマットに反映することにより適応する。現在のメモリ参照は16ビット値に関するので、XOR 6 は図５で説明したように実行され、オフセット６から正しい大エンディアン半ワードを取出す。

【００４０】アドレスオフセット変更の上記説明は16ビット、32ビット、64ビットのワードサイズでは正しく、そして他のワードサイズに容易に拡張することが分かる。例えば、128 ビットワードサイズを有するプロセッサは第１のステップで16バイトワード反映を実行する。この場合、バイト０はバイト15と交換され、バイト１はバイト14と交換され、バイト２はバイト13と交換され、バイト３はバイト12と交換され、バイト４はバイト11と交換され、バイト５はバイト10と交換され、バイト６はバイト９と交換され、そしてバイト７はバイト８と交換される。第２のステップの間に、プロセッサは16進数で８ビット参照のXOR F、 16ビット参照のXOR E、 32ビット参照のXOR C、 64ビット参照のXOR 8 及び128 ビット参照のXOR0 を用いて、アドレスオフセットの最下位４ビットのXOR 演算を実行する。

【００４１】混合エンディアンコンピュータシステムの設計に含まれた１つの複雑さはキャッシュメモリ管理に関連した問題点である。この問題点は、第２のタスクのエンディアンの下のキャッシュにまだ存在する間に、第１のタスクのエンディアンの下に取出されたデータが再使用されるときに起きる。例えば、データが小エンディアンの規則の下に取出された場合、キャッシュメモリ内のデータは今度のアドレス変更の準備ができて反映されている。しかしながら、データが大エンディアンの規則の下に取出された場合には、データは反映されていない。プロセス切換え時間で反映が行われたか否かを判定する容易な方法は既存のハードウェアにはない。

【００４２】１つの直観的な解法はタスクが変わる毎に単にキャッシュをフラッシュすることかも知れない。しかしながら、キャッシュのフラッシングはタスクを変える回数によりコンピュータシステムの性能に悪い影響を及ぼし、新しいタスクが代わりのエンディアンに属し、何か予測し制御することが困難なものが生ずることがある。本発明は、タスクエンディアンが変わる頻度を感知しないので、タスクエンディアンが変わる毎にキャッシュをフラッシュするよりもすぐれている。更に、行サイズ及び結合性のようなキャッシュ実現要素に基づいて、キャッシュをフラッシュする性能費用が変化することがある。従って、本発明が各タスクエンディアンが変わる毎にキャッシュをフラッシュするよりもすぐれているのは、本発明はキャッシュ設計の選択の可能性を制限しないためである。

【００４３】本発明は既存のIBM ２エンディアンパワーPCコンピュータシステムを高機能にするものである。タスク毎に反映及びアドレス変更を制御し追跡するためにプロセッサに混合エンディアン回路が付加されている。実行中のタスクがメインメモリからキャッシュメモリに

情報をロードすると、本発明の混合エンディアン回路はタスクのエンディアンを情報と共にキャッシュ行に記憶する。そしてタスク（即ち、同じタスク又は異なるタスク）がキャッシュメモリからの情報の取出しに移ると、本発明の混合エンディアン回路は、取出し中のタスクのエンディアンが情報のエンディアンに一致するかどうかを判定する。もし一致すれば、通常の取出しが許される。本発明の混合エンディアン回路がキャッシュミスを強制するのは、取出し中のタスクのエンディアンが情報のエンディアンと一致しない時のみである。キャッシュミスの後、情報は適切なフォーマットでキャッシュメモリにロードされ、そして実行中のタスクにより通常の方法で取出される。

【００４４】反映及びアドレス変更の制御及び追跡に加えて、本発明は割込み処理中の状況切換えを処理する機構も提供する。本発明の混合エンディアン回路は大及び小エンディアンタスクの両者が１つのコンピュータシステムに存在し且つタスク毎に実行することを可能にするから、そして割込み処理ソフトウェアはそれ自身が特定のエンディアンに属するから、本発明は割込みルーチンがそれら自身のエンディアンの下にプロセッサを制御しつつ、しかも延期されたタスクのエンディアンを保存することを可能にする手段を提供する。

【００４５】［詳細な説明］図１は本発明のコンピュータシステムのブロック図を示す。良好な実施例のコンピュータシステムは高機能のIBM AS/400中型コンピュータシステムである。しかしながら、本発明の機構及び装置は、どのコンピュータシステムにも、そのコンピュータシステムが複雑な多重ユーザ計算装置であるか、又は単一ユーザ装置、例えばパーソナルコンピュータないしはワークステーションであるかに関係なく、等しく当てはまることを当業者は理解するであろう。図１の分解組立図に示すように、コンピュータシステム100 はプロセッサ101 を有し、プロセッサ101はキャッシュメモリ103 及び混合エンディアン回路175 を介してシステムバス150 に接続され、そして混合エンディアン回路175 は反映回路121 を有する。メインメモリ 109、補助記憶装置インタフェース 140、端末インタフェース 145及び通信インタフェース 147もシステムバス150 に接続される。

【００４６】プロセッサ101 はアドレス変更回路119 及びマシン状態レジスタ(MSR)105を有する。プロセッサ101 は高機能のIBM ２エンディアンパワーPCプロセッサであるが、２エンディアンプロセッサはどの２ステップでも使用できる。アドレス変更回路119 は前述のアドレス変更を実行する責任を有するが、その機能はソフトウェアでも実現できる。MSR 105 は、プロセッサ101 に関連した現在のタスク情報を含むとともに、小エンディアン(LE)状況ビット129 及び割込み小エンディアン(ILE) 状況ビット127 も含む。LEビット129 はメモリ参照中に反映及びアドレス変更を実行すべきかどうかを示す。LEビット129 の値はオペレーティングシステム165 によりセットされ、プロセッサ101 で実行中の現在のタスクのエンディアンを反映する。本発明の混合エンディアン環境では、LEビット129 の値は、プロセッサ101 が異なるエンディアンのソフトウェアタスクを実行するとき実時間で変化する。

【００４７】ILE ビット127 は、LEビット129 がプロセッサ割込みを受取るときの状態を示す。ILE ビット127 はソフトウェア割込みハンドラとして選択されたエンディアンを反映する。アプリケーションプログラムエンディアン（即ち、大エンディアンプログラム155 又は小エンディアンプログラム 160）から割込み処理エンディアンに変更するとき、エンディアンの変更は、もしあれば、割込みプロセスの一部分でなければならない。割込みハンドラエンディアンの選択を反映するために、ILE ビット127 の値はコンピュータシステム100 の初期始動時にオペレーティングシステムにより一度だけセットすればよいことが望ましい。割込みハンドラのエンディアンバイアスは一般に初期始動後は変更されないので、ILE ビット127 の値は一般に実時間では変更されない。ILE ビット127 の値は一般性を失わずに一定値に固定できる。しかしながら、これは割込みハンドラを強制する副作用を有するので、システムソフトウェアはできれば特定のエンディアンに属すべきである。

【００４８】キャッシュメモリ103 は反映回路121 並びにキャッシュアレイ123 及び125 を有する。キャッシュメモリ103 は両方向結合、再コピーキャッシュであるが、本発明は特定のキャッシュ機構に限定されないことを当業者は理解するであろう。キャッシュアレイ123 及び125 の各々は複数のキャッシュアレイ素子を有する。技術的に知られているように、キャッシュアレイ素子の各々は実際のデータを含むキャッシュ行及び一定の制御情報を含む。

【００４９】反映回路121を含む混合エンディアン回路175は、本質的にパワーPCプロセッサ101 の２ステッププロセスを動的に制御し且つ追跡する責任を有する。しかしながら、本発明の意図及び範囲は、２エンディアン、パワーPCコンピュータシステムで用いられる、特定の２ステッププロセスに制限されないことを当業者は理解するであろう。実際、本明細書に開示された機構は、選択されたステップの数又は細部に関係なく、データ及びアドレスの調整のどれにも等しく使用できる。

【００５０】本明細書の概説セクション及び図４～図１０で説明したように、反映回路121はメインメモリ109 からキャッシュメモリ103 に読取られたデータの反映を実行できる。一般性を失わずに、キャッシュメモリ103 と関連したキャッシュメモリ制御装置（図示せず）で走行するソフトウェアで反映回路121 を実現できることが当業者には理解されるであろう。

【００５１】メインメモリ109 は大エンディアンプログ

ラム 155、小エンディアンプログラム 160、オペレーティングシステム 165、割込みハンドラ 171、大エンディアンデータ 177、小エンディアンデータ 179及びその他のプログラム（図示せず）を含む。大エンディアンプログラム 155は大エンディアンデータ 177を期待し且つそれにより動作するように設計されているプログラムであるのに対し、小エンディアンプログラム 160は小エンディアンデータ 179を期待し且つそれにより動作するように設計されている。しかしながら特別なケースでは、プログラムは代わりのエンディアンのデータを期待し且つそれにより動作するように設計できる。オペレーティングシステム 165は高機能のIBM マイクロカーネルに基づいた多重タスク処理オペレーティングシステムである。しかしながら、適切な多重タスク処理オペレーティングシステムはどれも用いてもよい。図１７に関連して後に詳細に説明されるように、ハードウェア割込み機構は本発明のILE ビット127 を利用するように機能が高められている。

【００５２】補助記憶装置インタフェース140 は、コンピュータシステム100 と補助記憶装置、例えば磁気又は光記憶装置とのインタフェースに用いられる。

【００５３】端末インタフェース145 は、システム管理者及びコンピュータプログラマが通常はプログラマブルワークステーションを介してコンピュータシステム100 と通信することを可能にする。通信インタフェース147 は、コンピュータシステム100 と外部通信ネットワーク、例えばローカルエリアネットワーク(LAN) 及び広域ネットワーク(WAN) とのインタフェースに用いられる。図１に示されたシステムは１つのメインCPU 及び１つのシステムバスのみを有するが、本発明は複数のメインCPU 及び複数のI/O バスを有するコンピュータシステムにも等しく用いられる。同様に、良好な実施例のバスは典型的なハードワイヤの多重ドロップであるが、両方向性通信をサポートするどの接続手段でも用いうる。

【００５４】［混合エンディアン動作］プロセッサ101 が走行している時、データ取出しを実行する通常のプロセスは図１１～図１６に示される。良好な実施例のプロセッサは高機能のIBM ２エンディアンパワーPCプロセッサであるから、これらの図面に示されたステップは大エンディアンの内部エンディアンバイアスを有するプロセッサについて記述される。図１１～図１６の以下の記述は本発明の混合エンディアンコンピュータシステムによるデータの取出し及び記憶の方法を説明する。この説明はデータ取出し、データ記憶及びキャッシュ再書込み動作に限定されているが、本発明の機構は容易に命令取出し動作の処理に拡張できることを当業者は理解するであろう。

【００５５】任意の取出し又はデータ記憶動作の前に、オペレーティングシステムは、LEビット129 の値をそれが実行中のタスクのエンディアンを表わすようにセットするようにプロセッサ101 に指示する。

【００５６】［データ取出し及び記憶動作（図１１のブロック605)］図１１及び図１２は混合エンディアン回路175 の論理流れ図である。これらの図面に記述された高レベルの論理的な流れを設計する同等な方法の数がいくつもあることを当業者は理解するから、本明細書では特定のハードウェア概要図については説明しない。図１１のブロック605 で、プロセッサ101 はメモリからの取出しを開始する。これはプロセッサ101 にキャッシュメモリ103 をアクセスさせる。プロセッサ101 によるキャッシュメモリ103 のアクセスから起こりうる３つのメモリ参照シナリオがある。しかしながら、図１１のブロック605 の処理は３つのシナリオ全てに共通であるから、ブロック605 の内部の働きの一般的な説明を最初に行う。

【００５７】図１３～図１６は図１１のブロック605 の分解組立概要図を示す。図１３及び図１４は、１つのコンピュータシステムで、異なる（即ち、大エンディアン対小エンディアン）情報フォーマット期待を有するタスクが共存し且つタスク毎に実行することを可能にするために用いられる混合エンディアン回路175 の重要な素子を示す。本発明の良好なキャッシュ管理ロジックの関連下位セットが示される。説明のために、図１３は両方向結合、再コピーキャッシュを示すが、本発明は一般性を失わずに他のキャッシュ設計に拡張しうることを当業者は認識するであろう。図１４はアドレス比較器660 を示す論理概要図である。並列アドレス比較器661 は同じ回路、出力 "ヒットミスLE$_i$(HitMissLE$_i$)" 及び "ヒット$_i$(Hit$_i$)"を有するが、図示されない。図１５は、図１４に示されたオブジェクトにより記述された、周知の排他的OR(XOR) 及び排他的NOR(XNOR) 論理演算の真値のテーブルを示す。これらのタイプの論理回路の構成は技術的によく知られている。図１６は図１４に示された比較器の出力の関連組合せを示す。

【００５８】図１３に示された処理は、アドレス変換ユニット640 に供給される仮想アドレスで開始する。仮想アドレスは、実行中のタスクがプロセッサ101 に取出しを指示するデータのアドレスである。本発明は特定の仮想変換機構のどれにもに限定されず且つ（仮想アドレス変換の完全な不在も一緒に含む）多くの仮想アドレス変換アーキテクチャをどれも使用できることを当業者は認識するであろう。アドレス変換ユニット640 の出力（即ち、変換実アドレス）はコピーされて変換実アドレスレジスタ642 に入れられる。そして変換実アドレスはハッシュユニット644 と比較器660 及び661 に供給される。

【００５９】ハッシュユニット644 は、変換実アドレス（即ち、変換実アドレスレジスタ642 の内容）を用いて、キャッシュメモリ103 のどの２つのアレイ素子がレジスタ642 に現に保持されているアドレスのキャッシュアレイ素子を保持しうるかを決定する。本発明の目的のために適切なハッシング機構はどれでも使用できるか

ら、本明細書にはハッシング機構の詳細は示さない。

【００６０】図１４は比較器660 の分解組立図を示す。冗長な説明を避けるために、同一の回路を有する比較器661 の詳細な表示又は説明はしない。しかしながら、比較器660 の回路により実行される動作は比較器661 の回路でも実際に並行して行われることを当業者は理解するであろう。ハッシュユニット644 により識別された２つのアレイ素子の１つ（例えば、646)に関連したタグ（例えば、タグ648)の値は、ビット毎に、比較器660 の排他的NOR 回路665 に転送される。タグ648 又は649 からの対応するアドレスビットは、図１３のレジスタ642 に含まれた変換実アドレスからの対応するアドレスビットと排他的 "NOR"される。これらの論理演算の結果はAND ゲート667 に供給される。AND ゲート667 の出力が１である場合、それは、対応するキャッシュアレイ素子にあるアドレスがレジスタ642 に含まれた変換実アドレスのアドレスに一致することを意味する。この場合、タグのアドレスビット０はレジスタ642 内のラッチされたアドレスのアドレスビット０に対応し、ビット１はビット１に対応する。以下、ハードウェア実現で一意的な実アドレスを決めるのに必要なビット数について同様に対応する。もちろん、これはプロセッサ101 が要求した情報がキャッシュメモリ103 に存在することを意味する。しかしながら、必要な情報がキャッシュメモリ103 に存在することは、該情報が実行中のタスクが期待した（即ち、反映された又は反映されない）形式で配列されていることを意味しない。

【００６１】キャッシュメモリ103(即ち、ハッシュユニット644 により識別されたキャッシュアレイ素子の１つ）にある情報が期待されたフォーマットで記憶されているかどうかを判定するために、LEビット129 及び記憶されたLE(RLE) ビット650 が用いられる。アドレスが比較されている２つのキャッシュ行にあるRLE ビット650 及び651 (RLEビット651 は図示せず）がセットされ、２つのキャッシュ行の現在の構成部が初期化され検査されたとき、セットされた制御行の１つとしてキャッシュアレイに記憶される。記憶された値は、キャッシュ行が初期化されたときのMSR のLEビットの値である（即ち、キャッシュ行に現に存在するデータのフォーマットに対応する)。 これらの記憶されたビットは本発明にとって重要である。なぜなら、それらは混合エンディアン回路175 が記憶され、そしてキャッシュアレイ素子が最初にロードされたとき反映が実行されたかどうかの説明を可能にするからである。

【００６２】次の説明から分かるように、RLE ビットの目的は、キャッシュメモリ103 のキャッシュアレイ素子にプロセッサ101 が要求したデータが実際に存在するが、それにもかかわらず間違って反映したフォーマットであるとき本発明が検出し管理できることである。この情報を得るために、RLE ビット650 は671 で排他的NOR され、そして現在のMSR のLEビット（即ち、LEビット129)と669 で排他的ORされる。AND ゲート667 及び排他的OR回路669 の出力はAND ゲート673 でAND され、そしてヒットミスLE_0(HitMissLE_0)677を生ずる。AND ゲート667 の出力はAND ゲート675 で排他的NOR ゲート671 の結果とAND され、そしてヒット$_0$(Hit_0)679を生ずる。値ヒットミスLE_0(HitMissLE_0)，ヒット$_0$(Hit_0)，ヒットミスLE_1(HitMissLE_1)及びヒット$_1$(Hit_1)は図１１のブロック605 の出力である。

【００６３】図１６のテーブルはヒットミスLE_0(HitMissLE_0)，ヒット$_0$(Hit_0)，ヒットミスLE_1(HitMissLE_1）及びヒット$_1$(Hit_1)をそれらの意味とともに示す。テーブル内の"X"は "ドントケア"の値を示す。685 に示すように、全ての値が０であることはプロセッサ101 が要求した情報がキャッシュメモリ103 に存在しなかったことを意味する、即ち通常のキャッシュミスが起きたことを意味する。687 及び689 に示すように、ヒット$_0$(Hit_0）又はヒット$_1$(Hit_1）の論理値が１であり且つヒットミスLE_0(HitMissLE_0）及びヒットミスLE_1(HitMissLE_1）の論理値が０であることは、プロセッサ101 が要求したデータが正しいフォーマットでキャッシュメモリ103 に実際に存在することを意味する。691 及び693 に示すように、ヒットミスLE_0(HitMissLE_0）及びヒットミスLE_1(HitMissLE_1）の論理値１は、プロセッサ101 が要求したデータが実際にキャッシュメモリ103 に存在するが、そのフォーマットは間違っていることを意味する。

【００６４】［シナリオ１：データが正しいフォーマットで存在する］このシナリオでは、ヒット$_0$(Hit_0）又はヒット$_1$(Hit_1）のどちらかが論理値１を有し且つヒットミスLE_0(HitMissLE_0）及びヒットミスLE_1(HitMissLE_1）はともに論理値０を有する。これはプロセッサ101 が要求したデータが実際にキャッシュメモリ103 に正しいフォーマットで存在することを意味する。これは、ブロック607(図１１）の結果がノー、ブロック611 の結果がイエスの場合である。従って、制御は図１２のブロック623 に移る。ブロック623 で、LEビット129 が検査され、現在のタスクが大エンディアンタスクであるか小エンディアンタスクであるかを判定する。繰り返して言えば、論理値１は現在のタスクが小エンディアンタスクであることを意味し、そして論理値０は実行中のタスクが大エンディアンタスクであることを意味する。LEビット129 が０である場合、データが直にキャッシュメモリ103 にコピーされたと想定されるので、アドレス変更は起きない。取出しはブロック625 で終了する。

【００６５】［シナリオ２：データが間違ったフォーマットで存在する］このシナリオでは、ヒットミスLE_0(HitMissLE_0）又はヒットミスLE_1(HitMissLE_1）は論理値１を有する。これは実行中のタスクについてプロセッサ101 が要求したデータが実際にキャッシュメモリ103 に存在するが、間違ったフォーマットであることを意味す

る。もちろん、これは主題のキャッシュアレイ素子がフラッシュされて適切なデータが適切なフォーマットでキャッシュメモリ103 に取り込まれる必要があることを意味する。（繰り返して言えば、適切な形式はタスクの要求に従って反映され又は反映されないことを意味する。）これはブロック607の結果がイエスでの場合であり、制御の流れがブロック609 に移る。ブロック609 は対応する（即ち、ヒットミスLE$_0$(HitMissLE$_0$）又はヒットミスLE$_1$(HitMissLE$_1$）のどちらが論理値１を有するかにより）アレイ素子の変更ビット（例えば、図１３の変更ビット658)がセットされているかどうかを判定する。論理値１に等しい変更ビットは、キャッシュメモリ103 のキャッシュアレイ素子にあるデータが前の記憶により変更されており、メインメモリ109 内の対応するデータはもはや最新ではないことを意味する。もしそうなら、関連したRLE ビットがブロック627 で検査される。RLE ビットがセットされている場合、キャッシュデータは反映され（即ち、最初の反映を逆転し）、そしてメインメモリ109 に再びコピーされる（ブロック629)。RLE ビットがセットされていない場合には、反映は不要であるから、キャッシュデータは直にメインメモリ109 にコピーされる（ブロック631)。ひとたびキャッシュデータが再びメインメモリ109 にコピーされれば、キャッシュアレイ素子はブロック633 で無効にされる。

【００６６】キャッシュアレイ素子が適切にフラッシュされた後、図１２のブロック613 でデータ参照が続く。ブロック613 で、LEビット129 が検査され、現在のタスクが大エンディアンタスクであるか小エンディアンタスクであるかを判定する。現在のタスクが大エンディアンタスクである場合、ブロック617 で、プロセッサ101は要求したデータをキャッシュメモリ103 にコピーし、そしてブロック625 に進んでデータを取出す。しかしながら、実行中のタスクが小エンディアンタスクである場合、プロセッサ101 は反映回路121 で反映を実行し（ブロック615)、反映されたデータをキャッシュメモリ103 にコピーし（ブロック619)、アドレス変更回路119 によりアドレス変更を実行し（ブロック621)、そして最後に、ブロック625 で、キャッシュメモリ103 からデータを取出す。このシナリオでは本発明はフラッシュされたキャッシュアレイ素子を再使用することに注目されたい。

【００６７】本発明の機構はメインメモリ109 にあるデータの実際のフォーマットに関係なく適切に機能することに注目されたい。シナリオ２は１つの良い例である。例えば、大エンディアンタスク（即ち、大エンディアンプログラム155 の１つ）が、間違ったフォーマット（即ち、反映された形式）のデータの発見のためのみに、キャッシュメモリ103 に既に存在するデータをアクセスすることを試みたと想定する。更に、この大エンディアンタスクが小エンディアンデータを処理するように設計されたので、小エンディアンデータに内在するデータをそれが故意にアクセスし且つ真の小エンディアンフォーマットのデータを受取ることを期待すると想定する。しかしながら、データは、小エンディアンタスクの要求でキャッシュメモリ103 に既にロードされたので、反映された形式で現にキャッシュメモリ内にある。前述のように、本発明はキャッシュアレイ素子をフラッシュしてデータをメインメモリ109 から同じアレイ素子にデータを返送するが、（それは大エンディアンタスクの要求で取込まれていたので）この返送は未反映の形式で返送される。即ち、本発明の機構はメインメモリ109 に存在するデータの実際の形式と無関係であるので、この特定の大エンディアンタスクが小エンディアンデータを期待しても本発明の機構にとって問題ではない。本発明のこの態様は交差エンディアンデータアクセス又は交差エンディアンデータ共有と呼ばれ、そして本発明は、アプリケーション自身のエンディアン規則、たとえ単一エンディアンマシン上でも真であると期待されたマシンモデルの下では、ハードウェアの取出し及び記憶が常にそうすると仮定すれば、１つのエンディアンでアプリケーションが対抗するエンディアンに内在するデータを処理する規則を正確に保持する。

【００６８】［シナリオ３：データは存在しない］このシナリオでは、ヒットミスLE$_0$(HitMissLE$_0$)，ヒット$_0$(Hit$_0$)，ヒットミスLE$_1$(HitMissLE$_0$）及びヒット$_1$(Hit$_1$）の論理値は全て０である。それはプロセッサ101 が要求した情報がキャッシュメモリ103 に存在しないことを意味する。ハッシュユニット644 により識別されたキャッシュアレイ素子は（シナリオ２のように間違ったフォーマットで）取出される情報を含まないから、このシナリオは、実際にフラッシュされる素子ではありえないほかは、上記のシナリオ２と同じように処理される。

【００６９】この点で、データをキャッシュメモリ103 に、そして最終的にはメインメモリ109 に記憶する全ての３つのシナリオは、図１２のブロック625 がプロセッサ101 からキャッシュメモリ103 にデータを書込む適切なロジックと置き換えられるほかは、前述の取出しシナリオと同じロジックを用いて実行される。

【００７０】I/O DMA 及び多重プロセッサメモリアクセスを含む、キャッシュ付きコンピュータシステムに共通の他の取出し及び記憶動作は記述されていないが、本発明の機構は容易に拡張され、それらを含みうることを当業者は理解するであろう。

【００７１】［割込み処理］前述のように、現在のソフトウェアタスクのエンディアンが小エンディアンである場合は、LEビットは常にセットされ、そして現在のソフトウェアエンディアンが大エンディアンである場合には、常にクリアされる。

【００７２】初期化時に、コンピュータシステム100 は既知のエンディアンで開始する。初期化の間、オペレー

ティングシステムは一般にILE ビット127 をセットし、割込みハンドラ171 が要求したエンディアン（即ち、割込みハンドラ171 が大エンディアンタスクであるか小エンディアンタスクであるか）を指示する。ILE ビット127 が割込み処理を要求されるのは、プロセッサ101 がいつでも割込みを要求されることがあるからである。現在のソフトウェアタスクから割込みハンドラ171への状況切換えの間、割込みハンドラ171 のエンディアンと異なるエンディアンで現在のソフトウェアタスクが実行している場合に、問題が起きる。状況切換えの間、現在のソフトウェアタスクから割込みハンドラ171 にアトミック的に制御が引渡されるので、プロセッサ101 もデータエンディアン変換を実行すべきか否かをアトミック的に変更せねばならない。本発明のILE ビット127 は、割込みハンドラ171 がプロセッサ101 の制御を有する間に、プロセッサ101 が状況切換え中にメインメモリ109 にあるデータを正しく解釈することを可能にする。

【００７３】本発明の混合エンディアンコンピュータシステムの良好な割込み処理方法が図17に示される。プロセッサ101 はブロック703 で割込み要求を受取る。状況切換えの部分として、MSR 105 が一時的にロケーションに記憶される（ブロック705）。高度にパイプラインされたマシンでデータ取出し及び記憶がうまく変換されているが、実際の処理（例えば、図１２のブロック625 参照）は少なくとも１つの動作についてなお保留されていることがある。もしそうなら、プロセッサ101 は前記動作が終了するのを待つ（ブロック706）。ILE ビット127 はLEビット129 にコピーされる（ブロック707）。そして任意の前のメモリ動作はILE ビット127 がLEビット129 にコピーされる前に終了しているべきであり、そしてLEビット129の新しい値が次のメモリ取出しを制御しなければならないことにも再び注目されたい。即ち、LEビット129 の値の変更はアトミック的である。これは、２エンディアン及び非混合エンディアン規則を設計する設計者がプロセッサを、故意に又は偶然にこれらの混合エンディアン要求に適合しないように、ある経路で設計することがあるという重要な観察のほかは、現在のプロセッサにより共通に処理される他の典型的な割込みのための割込み処理に類似する。信頼できる混合エンディアンマシンは全ての経路及びキャッシュ最適化が上記のアトミック性要求に適合することを保証しなければならない。２エンディアンマシンは１つの良好に与えられた割込みについてのみこれらの要求に適合する必要がある。ひとたび割込みルーチンが走行すれば、取出し及び記憶はLEビット129 の新しい値を用いて上記のように（即ち、図１１及び図１２で説明したように）行われる。

【００７４】ブロック721 で割込み処理が終了したのち、ブロック723 でMSR 105 が復元される。そしてプロセッサ101 はブロック725 で未終了の取出し及び記憶の終了を待つ。そして現在のソフトウェアタスクの実行は状況切換え前の点で続行する。上記のように、LEの古い値から新しい値への変更はどれもアトミック的でなければならない。記憶されたMSR を復元し、割込まれた命令の割込み処理からアトミック的に再開始する割込み命令からの復帰があることが想定される。

【００７５】［一貫性のあるメモリイメージ］本発明の機構はメインメモリに一貫性のあるイメージを供給すると同時に混合エンディアン動作も可能にすることに注目すべきである。一貫性のあるメインメモリイメージのデータイメージは補助記憶装置で見つかったものと一貫性がある。一貫性のあるメモリイメージは多重プロセッサ環境で混合エンディアン動作を提供するのに重要である。多重プロセッサ環境はしばしば、単一の共有メインメモリを有する１つのコンピュータシステムで走行する複数のプロセッサを必要とする。時には、これらのプロセッサ、例えばサービスプロセッサはその製造業者が異なる。これは、プロセッサによって２エンディアン能力を有し、又はそれを持たない、即ち同一エンディアン能力のみを持ちうるものがあることを意味する。しかしながら、〝エンディアンであること〟に、即ち必要なプロセッサの能力に関係なく、メインメモリイメージが補助記憶装置のそれと一貫性があることに価値があるので、データは、それが種々のプロセッサで実行する種々のタスクによりアクセスされるとき、反映されたフォーマットであることは意外ではない。実際、全てのプロセッサが同じ種類であった場合でさえも、メインメモリをアクセスするとき期待すべきものを全てのタスクが知るように一貫性のあるメモリイメージを維持することがなお重要である。

【００７６】［一般的な適用可能性］重要なことは、上記の例は回転ディスクのような補助記憶装置に記憶されたデータの使用を要したが、本発明はその適用可能性がこのタイプの記憶装置に限定されないことである。実際、本発明は１つのコンピュータシステムで又はこのシステムに大エンディアン及び小エンディアンの両者のデータを記憶し又は伝達するどの手段にも等しく適用されることを当業者は認識し理解するであろう。

【００７７】例えば、本明細書に開示された混合エンディアンコンピュータシステムの実施例は、バス接続されたコンピュータシステム又はプロセッサ間でデータが引渡された環境でも同様に実行するのは、結局は、引渡されたデータが大エンディアンプログラム155及び(又は)小エンディアンプログラム160によりキャッシュメモリ103及びメインメモリ109にもロードされるためである。同様に、通信インタフェース147を介して外部のコンピュータネットワークから受取られたデータも大エンディアンプログラム155及び(又は)小エンディアンプログラム160によりキャッシュメモリ103及びメインメモリ109にロードされる必要があるであろう。

【００７８】図示を簡単にするために、この明細書に示

された整数データの例は位置合わせ済のデータのみを使用するが、本明細書に記述された着想及び手法は位置合わせ未了のデータ（例えば、奇数アドレス境界で始まる16ビット整数）に容易に拡張でき、そして位置合わせ未了のデータのケースは本発明の意図及び範囲内にあることを当業者は理解するであろう。

【００７９】［大エンディアン及び小エンディアンの命令］本明細書を通じて暗示されているが、本発明の機構を用いることにより、コンピュータシステム100 で異なるエンディアンのタスクをどちらも実行できることが明示されるべきである。パワーPC及び他の同様なコンピュータシステムでは、大エンディアンプログラムの命令は大エンディアンフォーマットであり、そして小エンディアンプログラムの命令は小エンディアンフォーマットであるから、小エンディアンタスクが本発明の高機能のパワーPCプロセッサで実行することを可能にするために、上記の反映及びアドレス変更が必要である。

【００８０】メモリの将来の見通しから命令が実際にはデータであると仮定すれば、大エンディアン及び小エンディアンの命令が本発明の機構により処理される方法は、大エンディアン及び小エンディアンのデータが本発明の機構により処理される方法と論理的に同じことを当業者は理解するであろう。従って、大エンディアン及び小エンディアンの命令の一般的な処理の細部は本明細書では繰り返さない。しかしながら、図１には示されないがパワーPCコンピュータシステムに含まれることが知られているコンピュータシステム100 の命令キャッシュが命令を処理するためにキャッシュメモリ103 の代わりに用いられることは示す価値がある。命令は常に１ワードのサイズであるから、図５に示されたXOR 4 アドレス変更は用いられたアドレス変更のみであることも示す価値がある。もちろん、この後者のアドレス変更ステートメントは実際に使用するものの１つであり、本発明が適用できるものの１つではない。たとえ命令のサイズが図５に示されたサイズよりも小さいか又は大きい場合でも、そしてたとえ命令が一定のエレメント、大エンディアン又は小エンディアンで用いられるように設計された場合でも、本発明は、同等のLEビットにより暗示されたエンディアンにあるデータのアクセスを除いて、異なるサイズの命令に等しく適用されることを当業者は理解するであろう。

【００８１】まとめとして、本発明の構成に関して以下の事項を開示する。

（１）　プロセッサと、前記プロセッサで、タスク毎に、直接に、大エンディアンタスクを実行する大エンディアンプログラム及び小エンディアンタスクを実行する小エンディアンプログラムを含むメモリとを備えるコンピュータシステム。

（２）　前記メモリは、前記大エンディアンタスクによりアクセスされる大エンディアンデータ及び前記小エンディアンタスクによりアクセスされる小エンディアンデータを更に含む、上記（１）に記載のコンピュータシステム。

（３）　前記小エンディアンデータは前記大エンディアンタスクによりアクセスされ、そして前記大エンディアンデータは前記小エンディアンタスクによりアクセスされる、上記（２）に記載のコンピュータシステム。

（４）　前記メモリは前記小エンディアンデータ及び前記大エンディアンデータを補助記憶装置から受取る、上記（３）に記載のコンピュータシステム。

（５）　前記メモリは前記小エンディアンデータ及び前記大エンディアンデータを外部通信ネットワークから受取る、上記（３）に記載のコンピュータシステム。

（６）　前記メモリは前記コンピュータシステムにバス接続されている別のコンピュータシステムから前記小エンディアンデータ及び前記大エンディアンデータを受取る、上記（３）に記載のコンピュータシステム。

（７）　前記大エンディアンプログラムは大エンディアンフォーマットの命令を含み、そして前記小エンディアンプログラムは小エンディアンフォーマットの命令を含む、上記（１）に記載のコンピュータシステム。

（８）　少なくとも１つの高機能２エンディアンプロセッサを有する複数のプロセッサと、大エンディアンタスク及び小エンディアンタスクがどちらも前記少なくとも１つの高機能２エンディアンプロセッサで、タスク毎に、直接に、実行されるとき、前記大エンディアンタスクを前記プロセッサで実行する大エンディアンプログラム及び前記小エンディアンタスクを前記プロセッサで実行する小エンディアンプログラムを含むメモリとを備え、前記メモリは、前記少なくとも１つの高機能２エンディアンプロセッサの代わりのエンディアンモードでデータを取出す間に、前記大エンディアンタスク及び前記小エンディアンタスクが前記少なくとも１つの高機能２エンディアンで行われるデータ及びアドレス調整を受けないように、一貫性のあるメモリイメージを提供する多重プロセッサコンピュータシステム。

（９）　前記データは、前記大エンディアンタスクでアクセスされる大エンディアンデータ及び前記小エンディアンタスクでアクセスされる小エンディアンデータを含む、上記（８）に記載のコンピュータシステム。

（１０）　前記データは、前記小エンディアンタスクでアクセスされる大エンディアンデータ及び前記大エンディアンタスクでアクセスされる小エンディアンデータを含む、上記（８）に記載のコンピュータシステム。

（１１）　従来の２エンディアンプロセッサで行われるデータ及びアドレス調整を追跡し且つ制御する混合エンディアン機構を備えるコンピュータシステム。

（１２）　前記従来の２エンディアンプロセッサは大エンディアンバイアスされ、そして前記従来の２エンディアンプロセッサが小エンディアンデータを処理できるよう

に前記データ及びアドレス調整が行われる、上記(１１)に記載のコンピュータシステム。
(１３)　前記従来の２エンディアンプロセッサは小エンディアンバイアスされ、そして前記従来の２エンディアンプロセッサが大エンディアンデータを処理できるように前記データ及びアドレス調整が行われる、上記(１１)に記載のコンピュータシステム。
(１４)　１つのプロセッサで直接に実行するために大エンディアンタスクを開始するステップと、前記プロセッサで直に実行するために前記大エンディアンタスクに割込み、そして小エンディアンタスクを直ちに開始するステップとを含むコンピュータでエンディアンタスクを実行する方法。
(１５)　前記大エンディアンタスクは大エンディアンデータをアクセスし、そして前記小エンディアンタスクは前記小エンディアンデータをアクセスする、上記(１４)に記載の方法。
(１６)　前記大エンディアンタスクは小エンディアンデータをアクセスし、そして前記小エンディアンタスクは大エンディアンデータをアクセスする、上記(１５)に記載の方法。
(１７)　補助記憶装置からメモリに前記小エンディアンデータ及び前記大エンディアンデータを受取る、上記(１６)に記載の方法。
(１８)　外部通信ネットワークからメモリに前記小エンディアンデータ及び前記大エンディアンデータを受取る、上記(１６)に記載の方法。
(１９)　前記コンピュータシステムにバス接続されている別のコンピュータシステムからメモリに前記小エンディアンデータ及び前記大エンディアンデータを受取る、上記(１６)に記載の方法。
(２０)　前記大エンディアンタスクは大エンディアンフォーマットの命令を含む大エンディアンプログラムの複数のバージョンを実行し、そして前記小エンディアンタスクは小エンディアンフォーマットの命令を含む小エンディアンプログラムの複数のバージョンを実行する、上記(１４)に記載の方法。

【図面の簡単な説明】

【図１】本発明のコンピュータシステムを示すブロック図である。
【図２】IBM パワーPCアーキテクチャで用いるデータワードサイズを示すデータ図である。
【図３】大エンディアン及び小エンディアンのデータフォーマットを示すデータ図である。
【図４】IBM パワーPCアーキテクチャでデータがいかに反映されるかを表わすテーブルを示す図である。
【図５】IBM パワーPCアーキテクチャで小エンディアンアドレスがいかに変更されるかを示すデータ図である。
【図６】２エンディアン、パワーPCコンピュータシステムで用いる２ステップの小エンディアンプロセスを介して８ビットデータアイテムがいかに取出されるかを示すデータ図である。
【図７】２エンディアン、パワーPCコンピュータシステムで用いる２ステップの小エンディアンプロセスを介して、位置合わせされた１６ビットデータアイテムがいかに取出されるかを示すデータ図である。
【図８】２エンディアン、パワーPCコンピュータシステムで用いる２ステップの小エンディアンプロセスを介して、位置合わせされた３２ビットデータアイテムがいかに取出されるかを示すデータ図である。
【図９】２エンディアン、パワーPCコンピュータシステムで用いる２ステップの小エンディアンプロセスを介して、位置合わせされた６４ビットデータアイテムがいかに取出されるかを示すデータ図である。
【図１０】２エンディアン、パワーPCコンピュータシステムで用いる２ステップの小エンディアンプロセスを介して、３２ビットの小エンディアン整数の下位の１６ビットをいかに取出しうるかを示す図である。
【図１１】良好な実施例の混合エンディアン回路を記述する高レベルロジックの流れ図である。
【図１２】良好な実施例の混合エンディアン回路を記述する高レベルロジックの流れ図である。
【図１３】良好な実施例の混合エンディアン回路の重要な素子を示す概要図である。
【図１４】良好な実施例の混合エンディアン回路の重要な素子を示す概要図である。
【図１５】図１４に示されたオブジェクトにより記述された、周知の排他的OR(XOR) 及び排他的NOR(XNOR) 論理演算の真値のテーブルを示す図である。
【図１６】図１３及び図１４に示された比較器の出力の関連組合せを示す図である。
【図１７】良好な実施例の混合エンディアン回路が良好な実施例の割込みハンドラーといかに対話するかを記述する高レベルロジック図である。

【符号の説明】

100　コンピュータシステム
101　プロセッサ
103　キャッシュメモリ
105　マシン状態レジスタ(MSR)
109　メインメモリ
119　アドレス変更回路
121　反映回路
123　キャッシュアレイ
125　キャッシュアレイ
127　ILE(状況)ビット
129　LE(状況)ビット
140　補助記憶装置インタフェース
145　端末インタフェース
147　通信インタフェース
150　システムバス

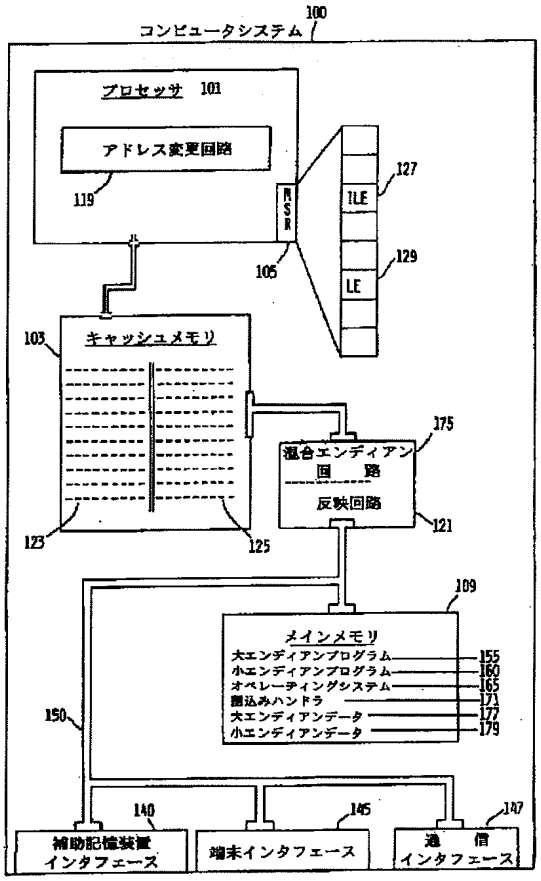
27

155	大エンディアンプログラム
160	小エンディアンプログラム
165	オペレーティングシステム
171	割込みハンドラ
175	混合エンディアン回路
177	大エンディアンデータ
179	小エンディアンデータ
640	アドレス変換ユニット

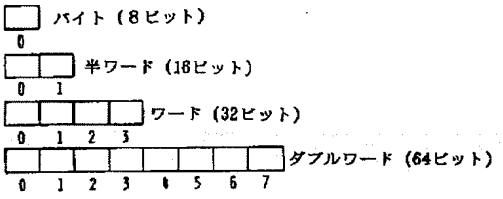
28

＊	642	変換実アドレスレジスタ
	644	ハッシュユニット
	660	アドレス比較器
	661	並列アドレス比較器
	665	排他的NOR回路
	667	ANDゲート
	671	排他的NOR回路
＊	669	排他的OR回路

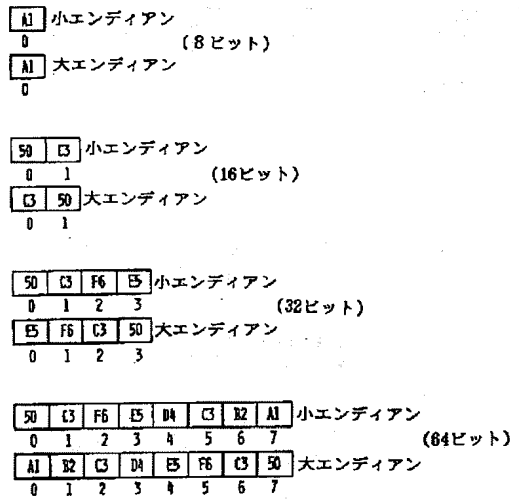
【図１】



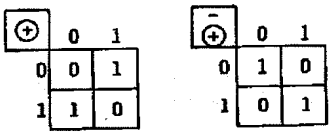
【図２】



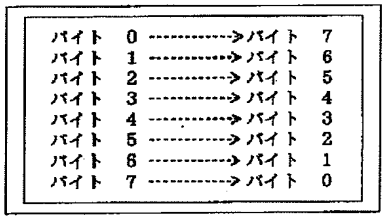
【図３】



【図１５】



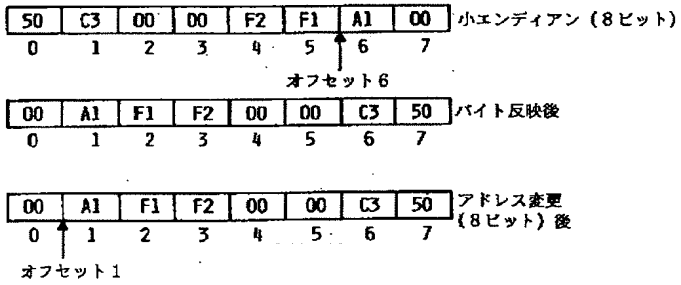
【図４】



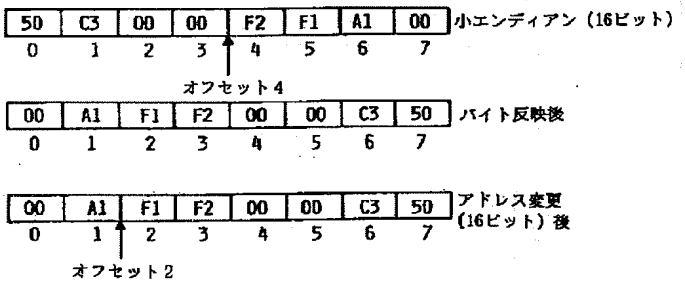
【図５】

ワードサイズ	アドレス変更
バイト（８ビット）	XOR 7
半ワード（16ビット）	XOR 6
ワード（32ビット）	XOR 4
ダブルワード（64ビット）	XOR 0

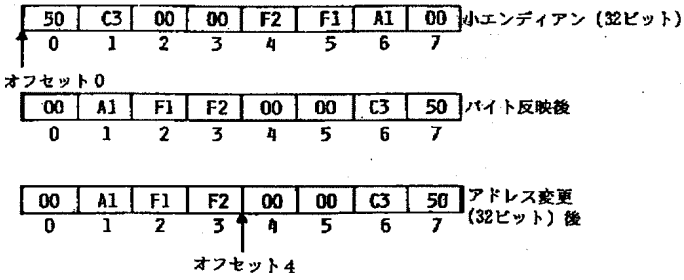
【図６】



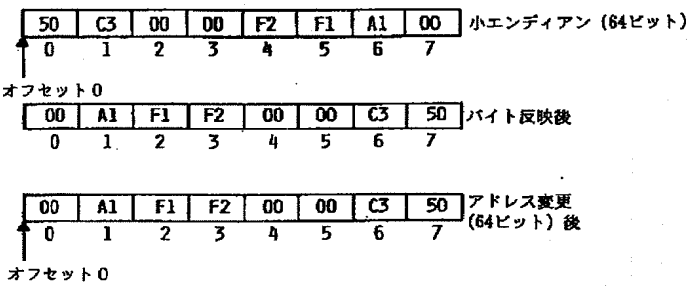
【図７】



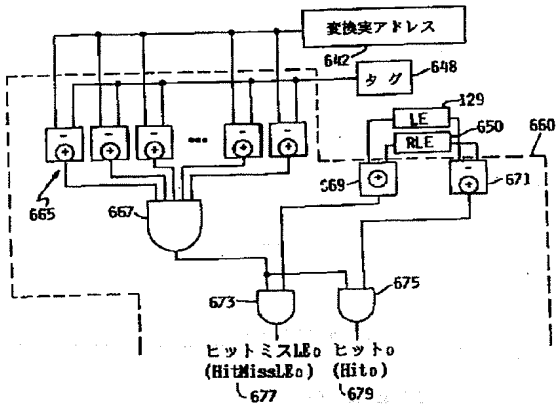
【図８】



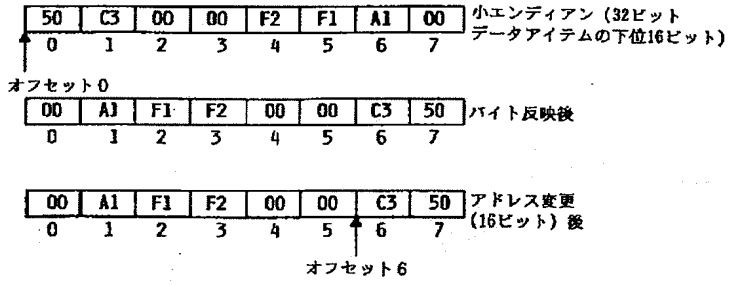
【図９】



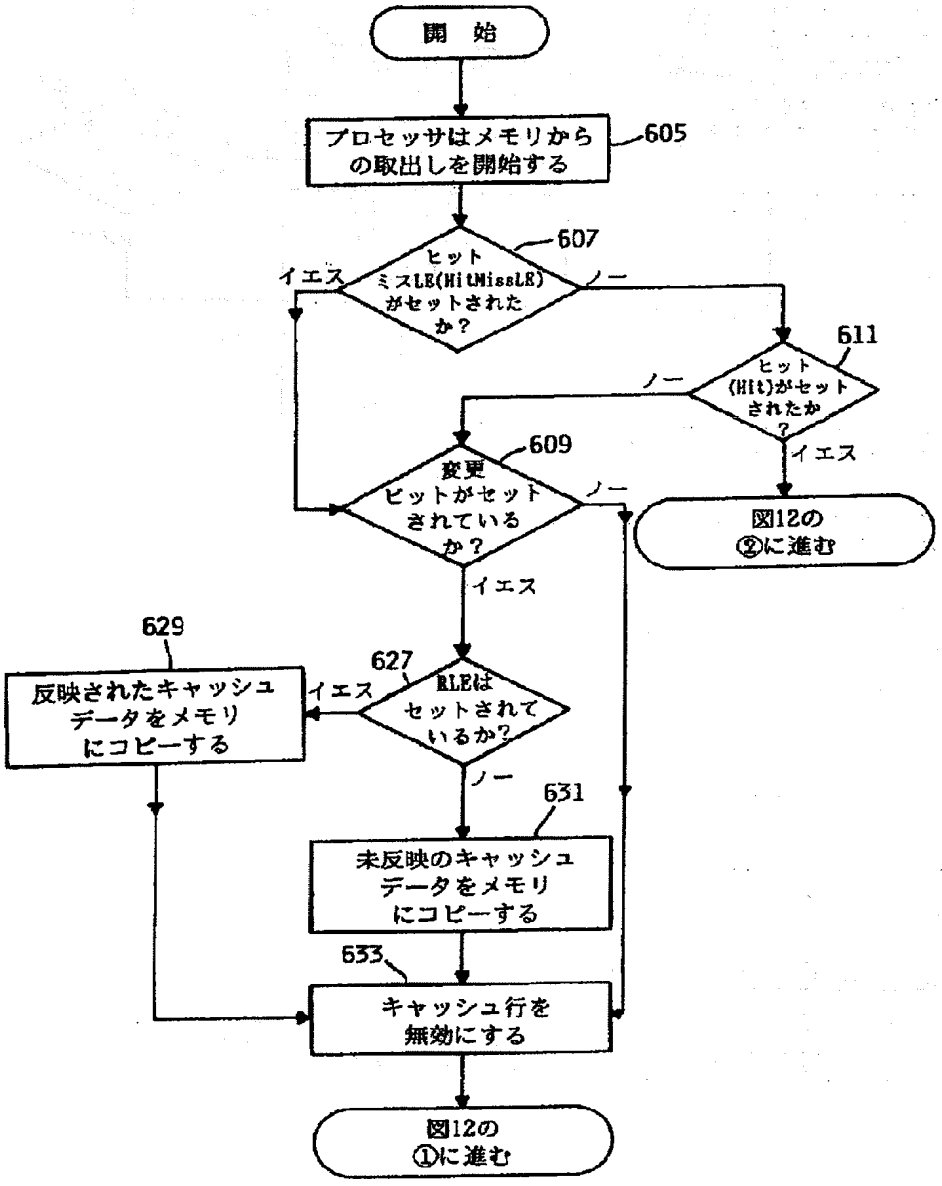
【図１４】



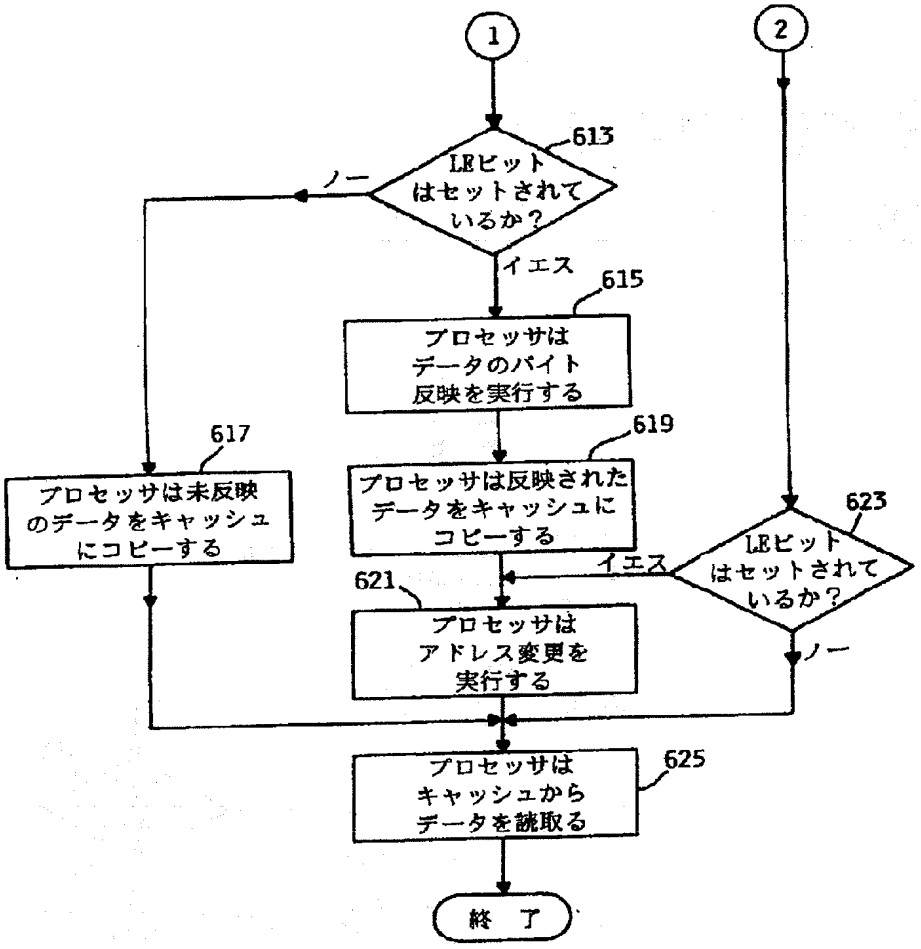
【図１０】


50 C3 00 00 F2 F1 A1 00
0 1 2 3 4 5 6 7
小エンディアン（32ビット
データアイテムの下位16ビット）
オフセット０
00 A1 F1 F2 00 00 C3 50
バイト反映後
アドレス変更
(16ビット）後
オフセット６

【図１１】


開　始
プロセッサはメモリから
の取出しを開始する
605
ヒット
ミスLE(HitMissLE)
がセットされた
か？
607
イエス
ノー
ヒット
(Hit)がセット
されたか
？
611
ノー
イエス
変更
ビットがセット
されている
か？
609
ノー
図12の
②に進む
イエス
629
反映されたキャッシュ
データをメモリ
にコピーする
627
RLEは
セットされて
いるか?
イエス
ノー
631
未反映のキャッシュ
データをメモリ
にコピーする
633
キャッシュ行を
無効にする
図12の
①に進む

【図１２】

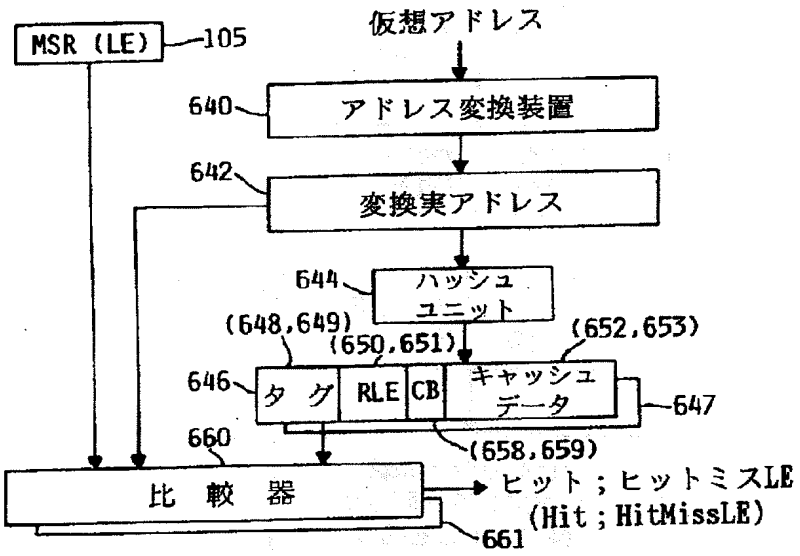


【図１６】

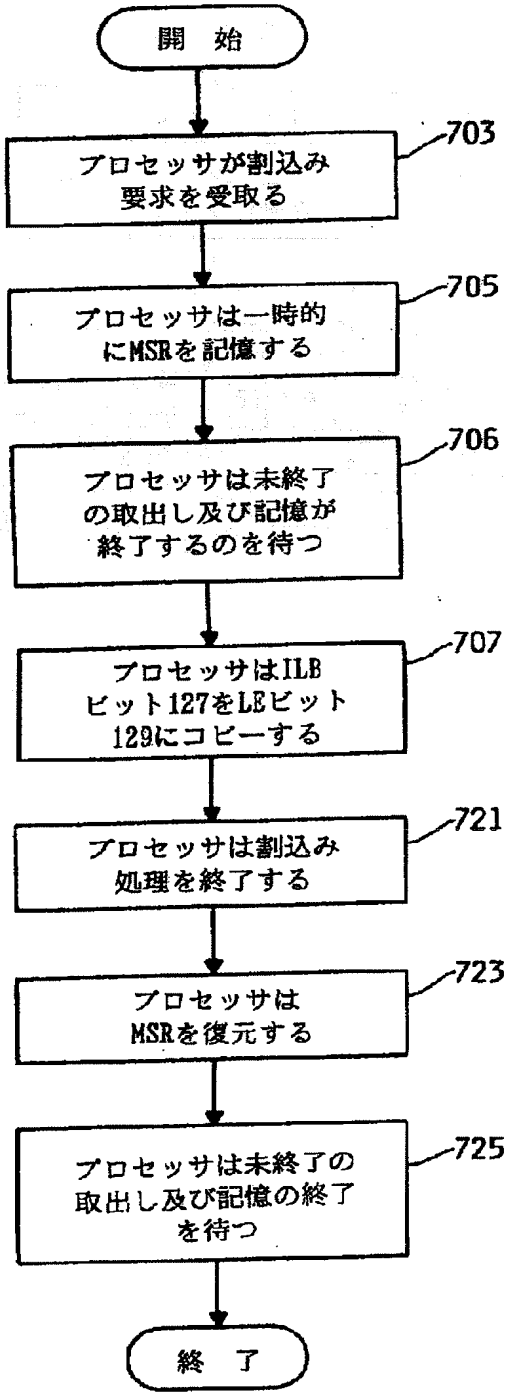
ヒットミスLE$_0$ (HitMissLE$_0$)	ヒット$_0$ (Hit$_0$)	ヒットミスLE$_1$ (HitMissLE$_1$)	ヒット$_1$ (Hit$_1$)
0	0	0	0
0	1	0	1
1	0	1	0
0	0	0	0

通常のキャッシュミス　←685
アレイ素子上のキャッシュヒット$_0$←687
アレイ素子上のキャッシュヒット$_1$←689
素子上のLE$_0$不一致←691
素子上のLE$_1$不一致←693

【図１３】


MSR (LE)
105
仮想アドレス
640
アドレス変換装置
642
変換実アドレス
644
ハッシュ
ユニット
(648,649)
(650,651)
(652,653)
646
タ グ
RLE
CB
キャッシュ
データ
647
660
(658,659)
比 較 器
ヒット；ヒットミスLE
(Hit；HitMissLE)
661

【図１７】


開　始
プロセッサが割込み要求を受取る
703
プロセッサは一時的にMSRを記憶する
705
プロセッサは未終了の取出し及び記憶が終了するのを待つ
706
プロセッサはILBビット127をLEビット129にコピーする
707
プロセッサは割込み処理を終了する
721
プロセッサはMSRを復元する
723
プロセッサは未終了の取出し及び記憶の終了を待つ
725
終　了

フロントページの続き

(72)発明者　マーチン・エドワード・ホプキンス
アメリカ合衆国ニューヨーク州、チャパクア、ダグラス・ロード　300
(72)発明者　ラリー・ウェイン・ロエン
アメリカ合衆国ミネソタ州、ロチェスタ、セナンドア・レーン　エヌ・ダブリュ　743
(72)発明者　エドワード・ジョン・シルハ
アメリカ合衆国テキサス州、オースティン、ピレニーズ　11509
(72)発明者　アンドリュウ・ヘンリー・ウォットレング
アメリカ合衆国ミネソタ州、ロチェスタ、マノー・ビュウ・ドライブ　エヌ・ダブリュ　4224